croc_code

Обезболивание enterprise-разработки

Применяем лучшее из мира микросервисов

Алексей Патрин

Почему enterprise и микросервисы?

- В последнее время разработка прикольных стартапов сократилась, а вот enterprise вполне себе набирает обороты
- А что-то хорошее должно случиться и в enterprise-разработке
- Поэтому используем микросервисную архитектуру

Что мы любим в микросервисах?

Разработка отдельных сервисов проще разработки большого приложения

Можно разрабатывать в несколько команд

Возможность выбора инструментов, наиболее подходящих для решения конкретной задачи

Модульность, DDD, SOLID

Масштабируемость решения в целом

Возможность частичного развёртывания

Что получаем в нагрузку?

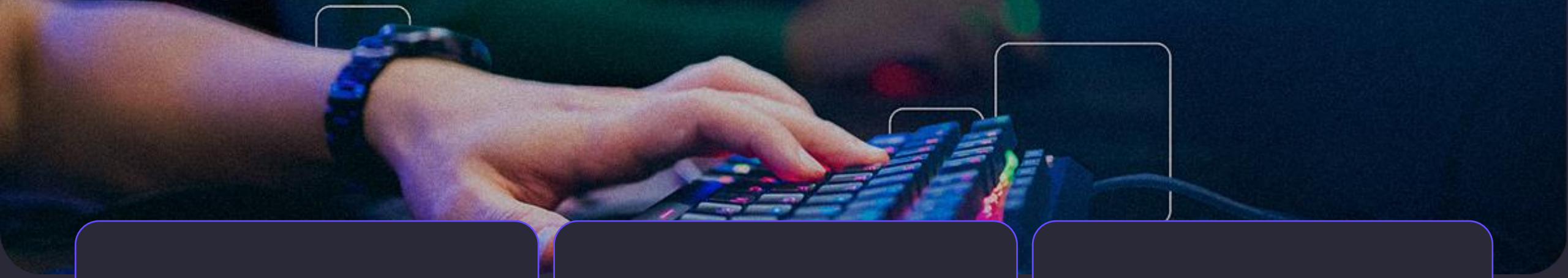
Инфраструктурные расходы при разработке каждого сервиса

Расходы на взаимодействие сервисов между собой

Необходимость в CI/CD

Надежная стыковка между собой разных версий сервисов

И так далее..



Команды

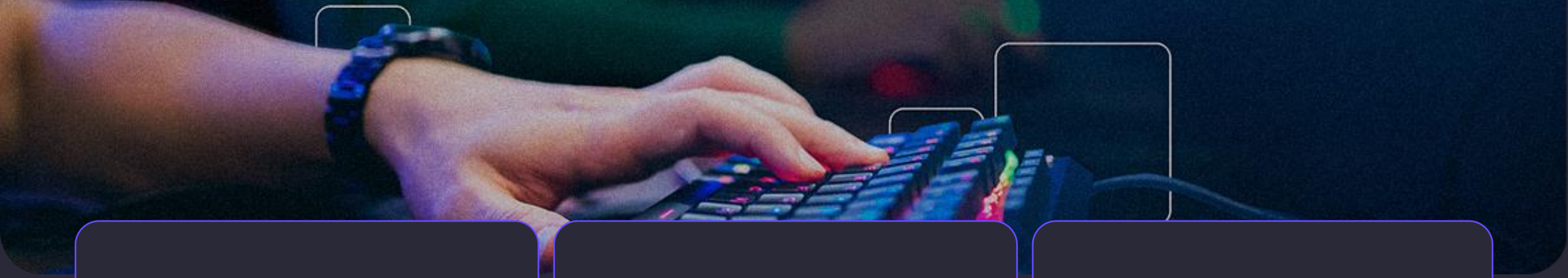
- Команда Ух
- Команда Ах
- Команда Эх

Логирование

- ...
- ...
- ...

...

- ...
- ...
- ...



Команды

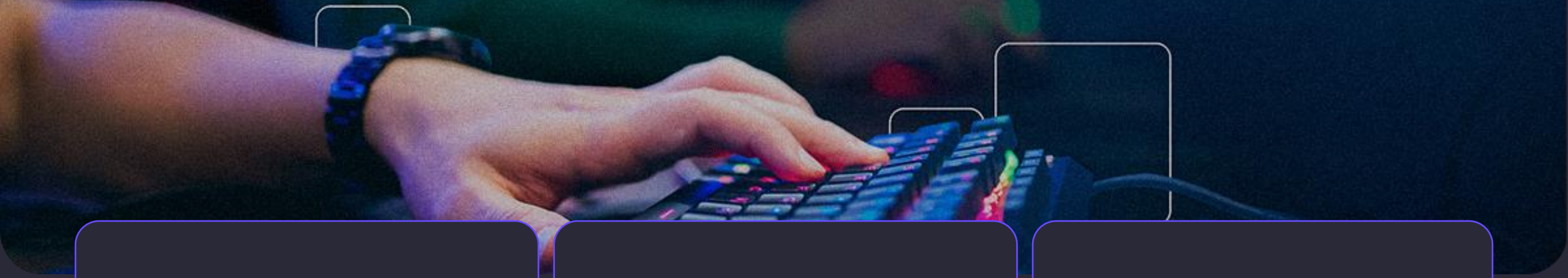
- Команда Ух
- Команда Ах
- Команда Эх

Логирование

- Файлы NLog.
- База данных Serilog
- ELK Logstash

Итог

- ...
- ...
- ...



Команды

- Команда Ух
- Команда Ах
- Команда Эх

Логирование

- Файлы NLog.
- База данных Serilog
- ELK Logstash

Итог

- Loki Serilog
- Loki Serilog
- Loki Serilog

Каркас микросервиса

Каркас микросервиса

Бизнес-логика

Каркас микросервиса

Бизнес-логика

- Логирование
- Версионирование
- Взаимодействие с другими сервисами
- Взаимодействие с Service Discovery
- То, что может пригодиться в нескольких сервисах

Всё ли хорошо в подходе с каркасом?

Всё ли хорошо в подходе с каркасом?

В действительности – breaking changes не такая уж и частая история. И чем дальше, тем менее частая, т.к. каркас «взрослеет» и стабилизируется

Всё ли хорошо в подходе с каркасом?

В действительности – breaking changes не такая уж и частая история. И чем дальше, тем менее частая, т.к. каркас «взрослеет» и стабилизируется

Функционал каркаса стоит использовать через интерфейсы и сервисы, подключаемые, например через DI. Даже каркас должен быть максимально изолированным и абстрактным

Всё ли хорошо в подходе с каркасом?

В действительности – breaking changes не такая уж и частая история. И чем дальше, тем менее частая, т.к. каркас «взрослеет» и стабилизируется

Функционал каркаса стоит использовать через интерфейсы и сервисы, подключаемые, например через DI. Даже каркас должен быть максимально изолированным и абстрактным

Каркас необходимо версионировать. Каждый сервис может работать со своей версией каркаса и обновлять каркас можно в каждом случае независимо.

Всё ли хорошо в подходе с каркасом?

В действительности – breaking changes не такая уж и частая история. И чем дальше, тем менее частая, т.к. каркас «взрослеет» и стабилизируется

Функционал каркаса стоит использовать через интерфейсы и сервисы, подключаемые, например через DI. Даже каркас должен быть максимально изолированным и абстрактным

Каркас необходимо версионировать. Каждый сервис может работать со своей версией каркаса и обновлять каркас можно в каждом случае независимо.

Каркас, и вообще всю разработку микросервисного приложения по возможности имеет смысл реализовывать только на одном языке/стеке. Почему? См. доклад Алексея Мерсона (https://techleadconf.ru/2022/abstracts/8931)

Логирование

Зачем?

Логирование

Зачем?

- Уменьшает объём отладки
- Найти ошибки на продакшене
- Позволяет провести разбор конфликтных ситуаций
- Это неплохое средство комментировать код

Логирование в микросервисной среде

Логирование в микросервисной среде

Логирование в единый получатель

- Сразу видна вся последовательность действий по всем сервисам

Логирование в микросервисной среде

Логирование в единый получатель

- Сразу видна вся последовательность действий по всем сервисам

Параметризация логов

- Поиск записей по разным критериям и их комбинациям

Логирование в микросервисной среде

Логирование в единый получатель

- Сразу видна вся последовательность действий по всем сервисам

Параметризация логов

- Поиск записей по разным критериям и их комбинациям

Примеры:

- Ошибка с идентификатором HTTP-запроса -> полные логи по данному запросу
- По идентификатору заявки –> полная история работы с заявкой

Инструменты для логирования

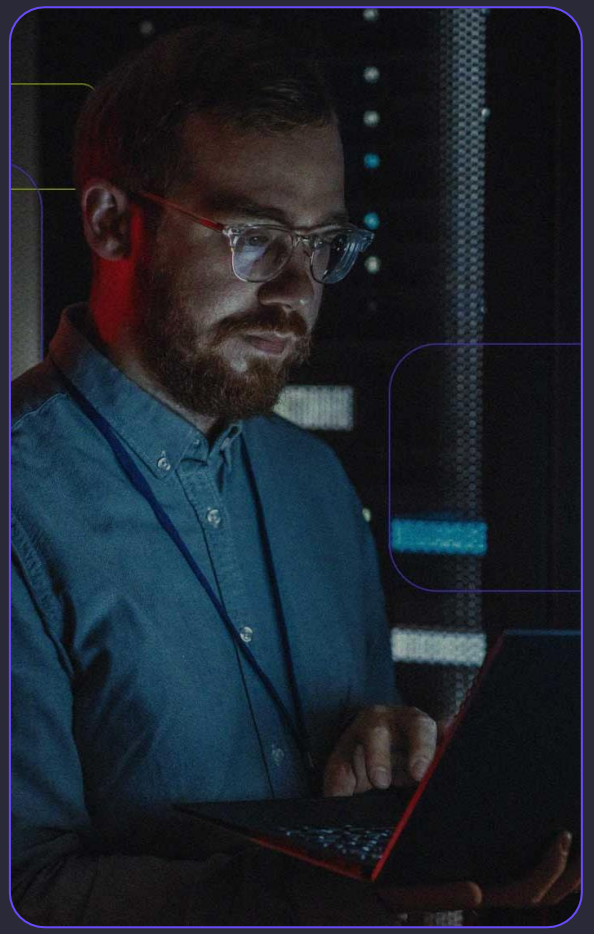
Библиотеки и фасады для логирования

- .NET - Microsoft.Logging
  - Serilog
  - Log4Net
  - NLog
- Java - SLF4J (Simple Logging Facade)
  - Log4J 2

Среда хранения и просмотра логов

- База данных
- ELK (Elastic search, Logstash, Kibana)
- Grafana, Loki, Promtail

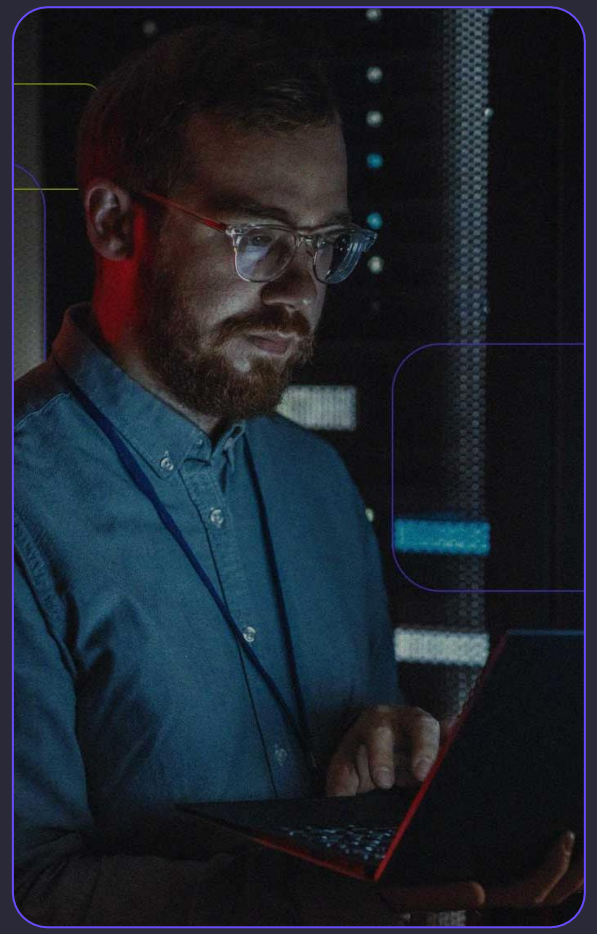
Версионирование API микросервисов



Версионирование API микросервисов

Версия API состоит из 3 чисел:

- Major – кардинальные изменения в API – полная несовместимость
- Minor – расширение API. Более старшая версия API может обслуживать клиентов, разработанных для более младшей версии
- Build – реализация API поменялась, само API не поменялось



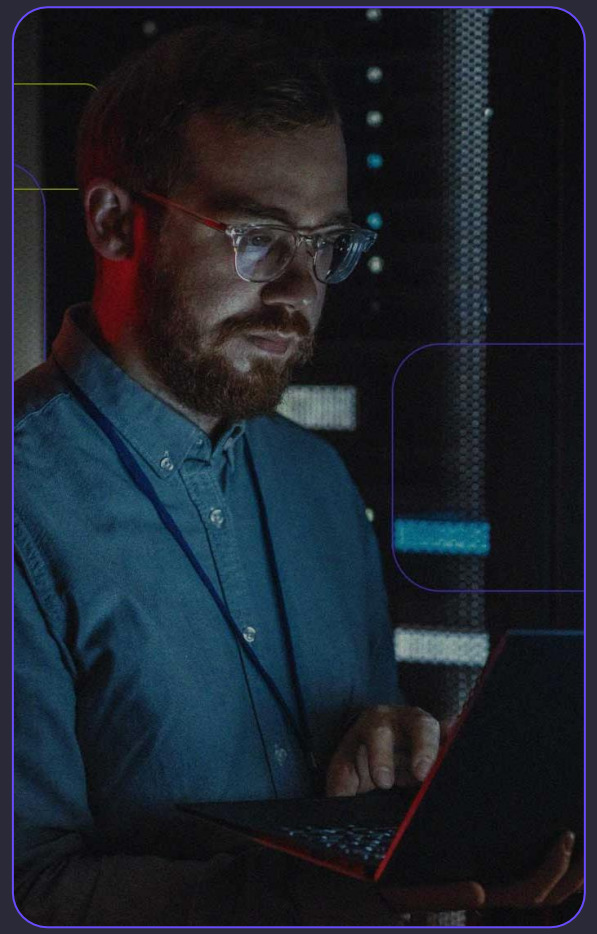
Версионирование API микросервисов

Версия API состоит из 3 чисел:

- Major – кардинальные изменения в API – полная несовместимость
- Minor – расширение API. Более старшая версия API может обслуживать клиентов, разработанных для более младшей версии
- Build – реализация API поменялась, само API не поменялось

Когда проверять версию API

- При взаимодействии
- При старте
- При деплое
- Новая версия по новому адресу
- UI – периодическая проверка



Взаимодействие между сервисами

REST и Websocket для Web-приложения

REST, GRPC, очереди для взаимодействия микросервисов между собой

Взаимодействие между сервисами

REST и Websocket для Web-приложения

REST, GRPC, очереди для взаимодействия микросервисов между собой

Ситуация: Один сервис-поставщик — много сервисов-потребителей

Разработчику каждого потребителю объяснять как взаимодействовать с сервисом-поставщиком?

Или заставить его читать документацию (которой обычно нет)?

Взаимодействие между сервисами

Как сервисам искать друг друга...

Идеальный вариант запуска сервиса в работу

Service Discovery

- Поиск адреса нужного сервиса по имени
- Health Monitoring
- Load balancing
- Централизованное конфигурирование

Service Discovery

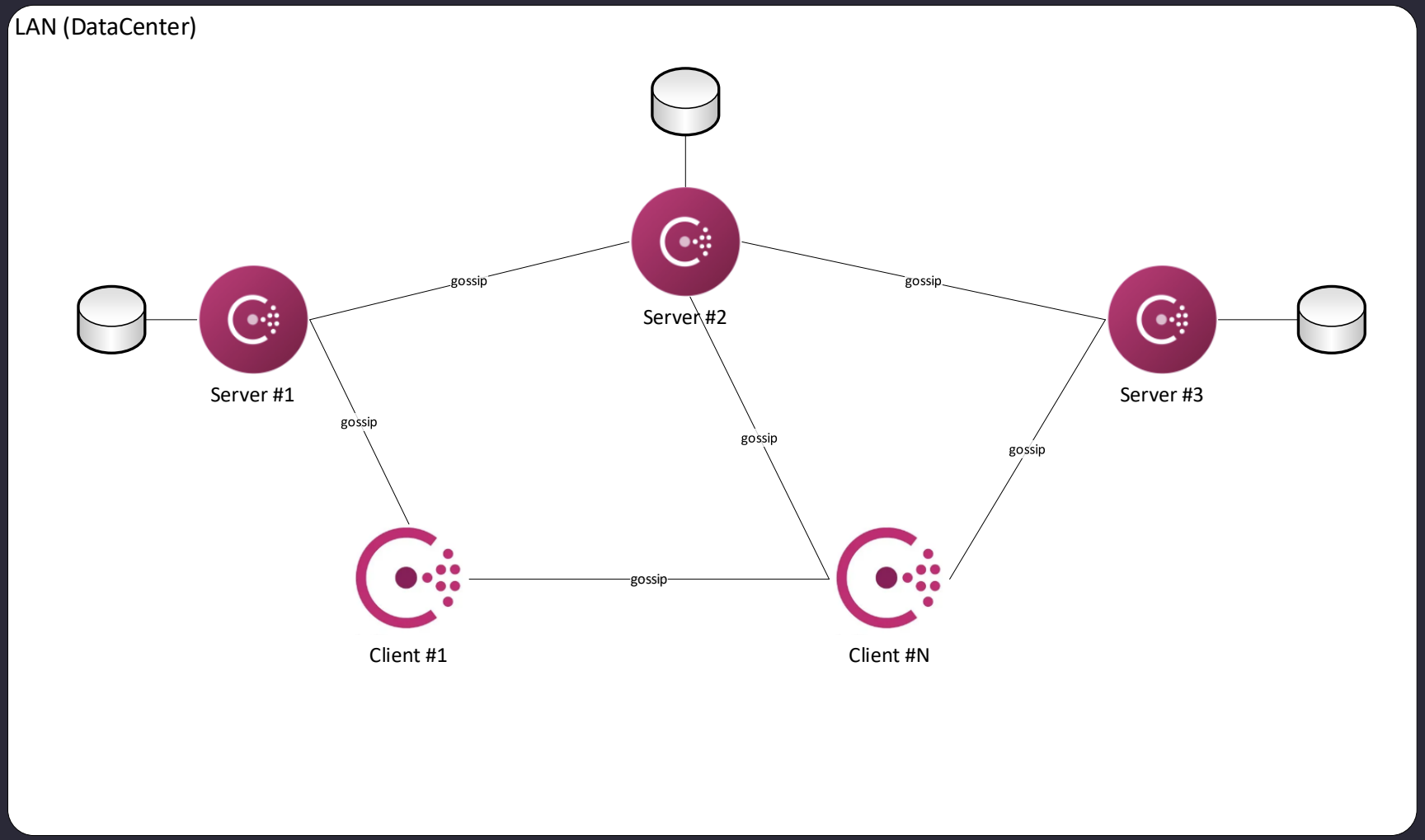
- Поиск адреса нужного сервиса по имени
- Health Monitoring
- Load balancing
- Централизованное конфигурирование

Kubernetes ??

Service Discovery / Consul

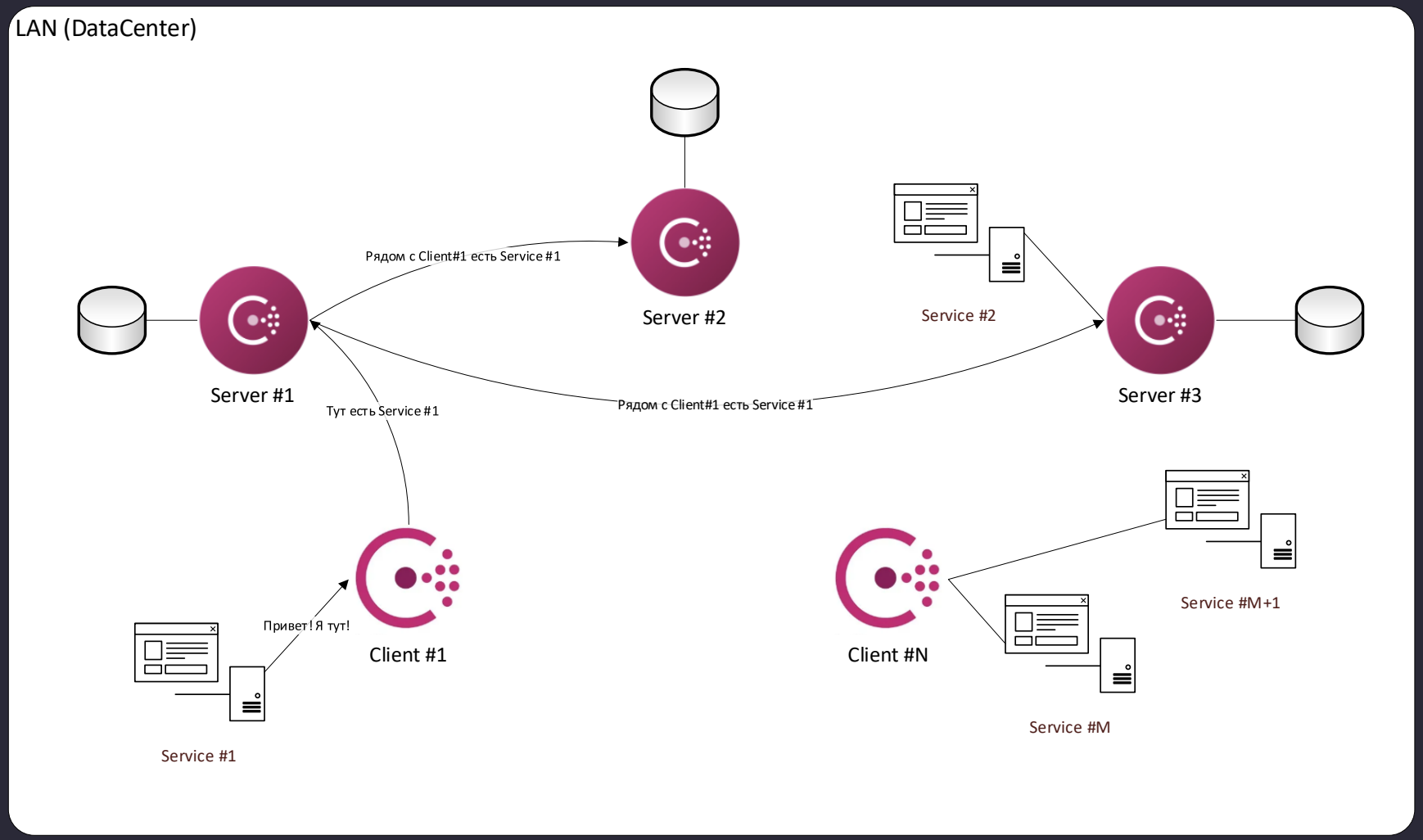
- Единственный исполняемый файл, запускаемый на каждом хосте (или поде)
- Каждый сервис взаимодействует только с локальным агентом
- Сервисы сами регистрируется в Consul через его API
- Адреса сервисов — через DNS или API
- Встроенный функционал Health Monitoring
- Встроенное хранилище Key/Value
- K/V можно расширить до хранения секретных данных с помощью Vault

Service Discovery / Consul / Как работает



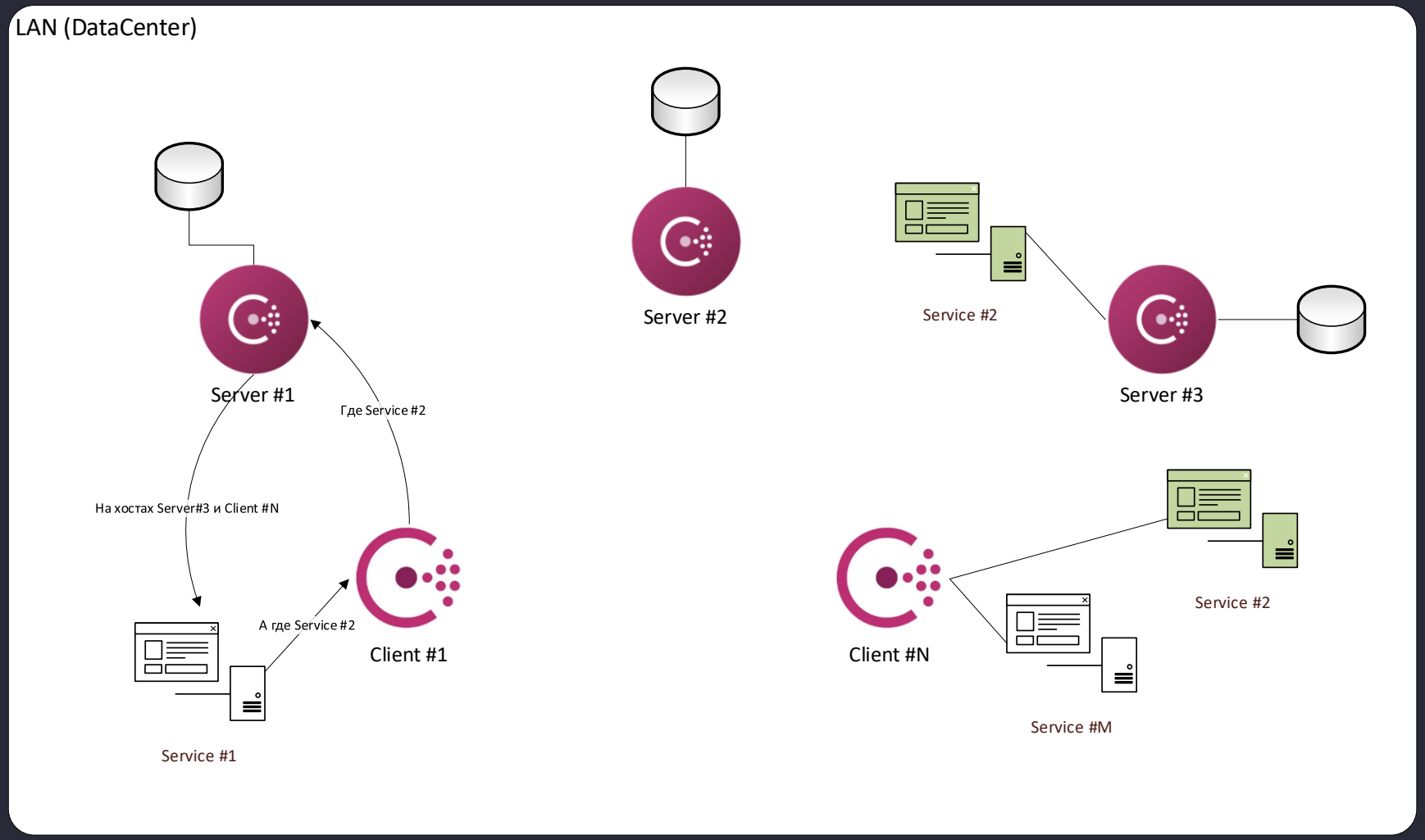
- На каждом хосте запускается агент Consul
- Каждый агент может быть сервером или клиентом
- Агенты обнаруживают друг друга в локальной сети по протоколу Gossip
- Среди серверов периодически проходят выборы и один из серверов становится лидером

Service Discovery / Consul / Регистрируем сервис



- Сервис при старте регистрируется в агенте Consul
- Если ближайший агент - клиент – регистрация перенаправляется на сервер.
- Сервера обмениваются информацией о зарегистрированном сервисе

Service Discovery / Consul / Ищем сервис



- Сервис запрашивает адрес сервиса по его имени в локальном агенте Consul
- Если локальный агент - клиент, запрос направляется на любой сервер.
- Сервер возвращает информацию из своей БД

Когда решил поделиться кодом...

Каталог сервисов компании

Разрабатывая сервис, определяя границы его ответственности - подумайте, возможно его можно будет применить в каком-то другом проекте

Переиспользовать сервис, как правило, проще, чем встроить к себе библиотеку. Особенно если библиотека написана на другом языке ☺

Стоит подумать на уровне компании о специальном ПО, предназначенном для каталогизации сервисов. Например - https://www.opslevel.com

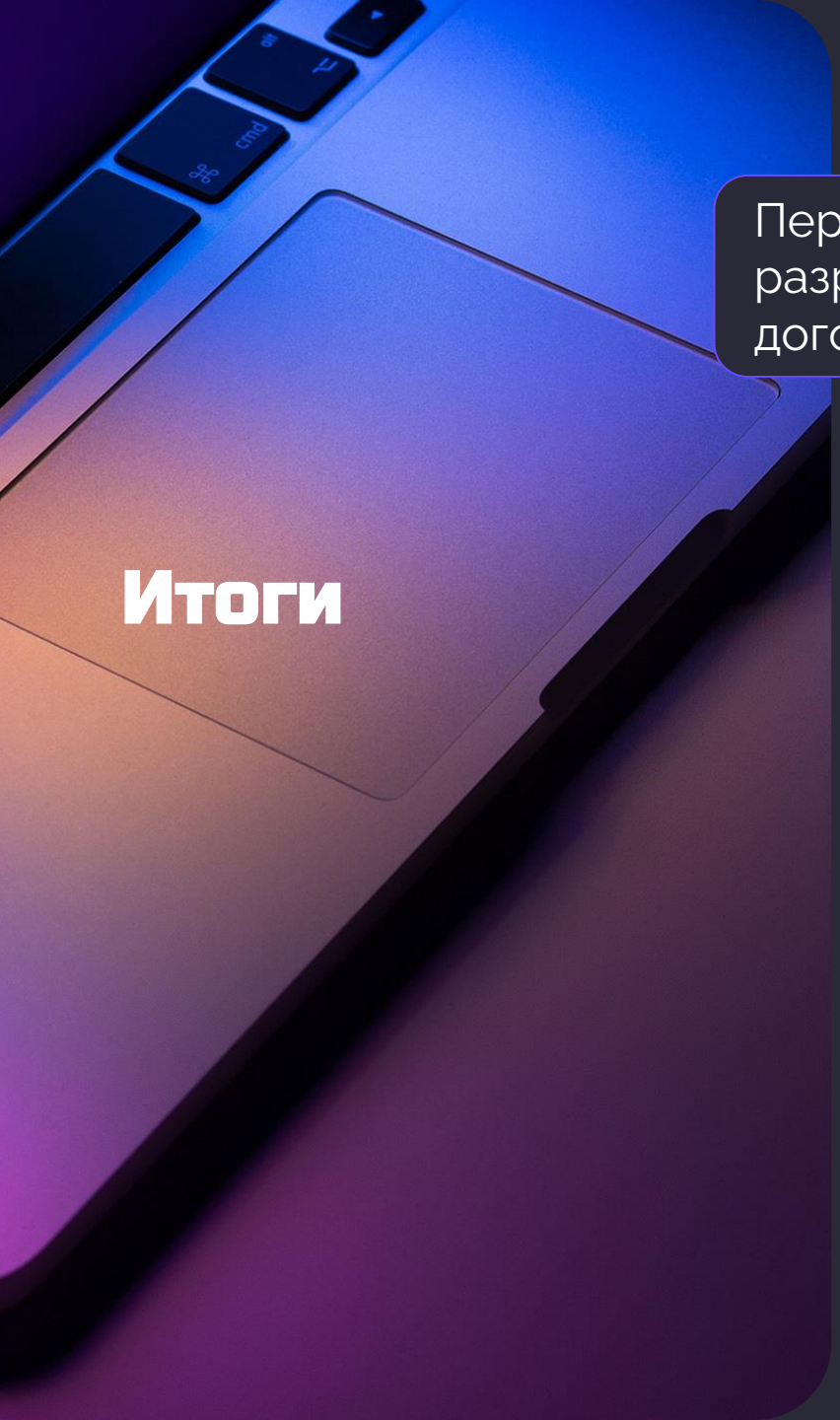
Каталог сервисов компании

Если существует интерфейсная библиотека — она может помочь в подключении сервиса

Каталог сервисов компании

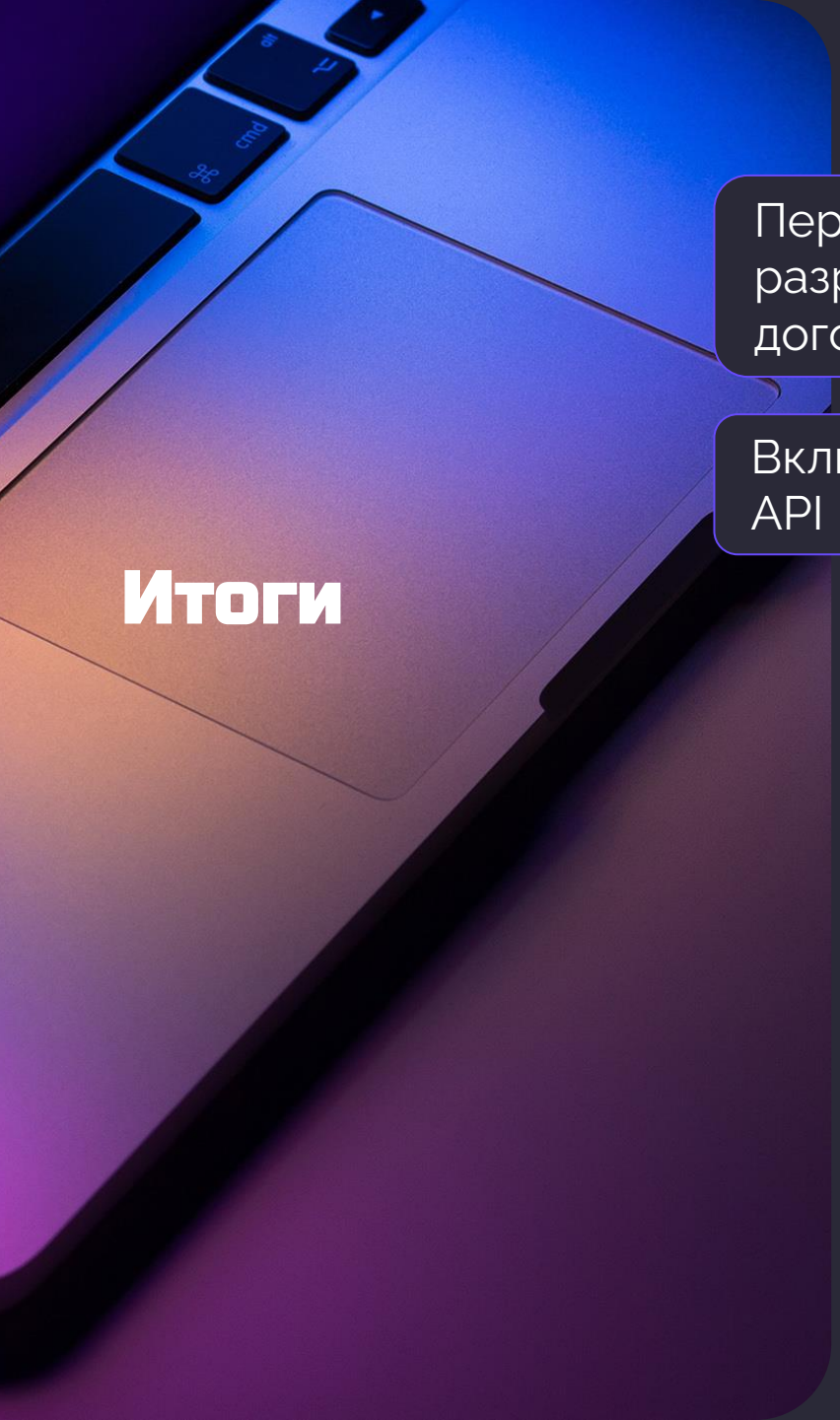
Если существует интерфейсная библиотека — она может помочь в подключении сервиса

Если сервис получает развитие в рамках другого проекта, то эти же плюшки может практически нахаляву получить исходный проект



Итоги

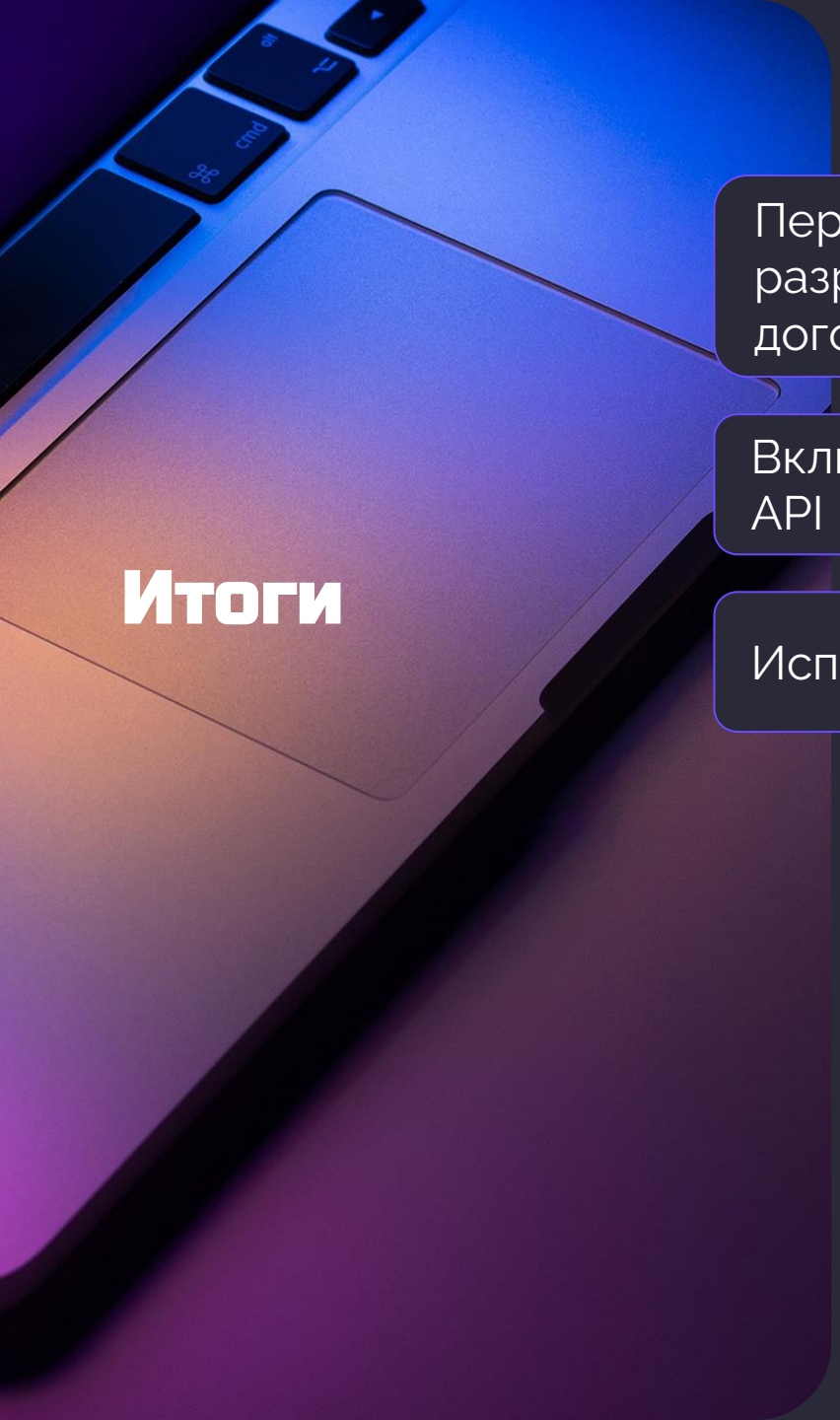
Перед разработкой проекта в микросервисной архитектуре разработайте каркас, где закрепите и развивайте все договоренности по разработке сервисов.



Итоги

Перед разработкой проекта в микросервисной архитектуре разработайте каркас, где закрепите и развивайте все договоренности по разработке сервисов.

Включите в каркас функции логирования и версионирования API



Итоги

Перед разработкой проекта в микросервисной архитектуре разработайте каркас, где закрепите и развивайте все договоренности по разработке сервисов.

Включите в каркас функции логирования и версионирования API

Используйте систему Service Discovery

Итоги

Перед разработкой проекта в микросервисной архитектуре разработайте каркас, где закрепите и развивайте все договоренности по разработке сервисов.

Включите в каркас функции логирования и версионирования

Используйте систему Service Discovery

Команда разработки сервиса предоставляет и сопровождает интерфейсные библиотеки к своему сервису

Итоги

Перед разработкой проекта в микросервисной архитектуре разработайте каркас, где закрепите и развивайте все договоренности по разработке сервисов.

Включите в каркас функции логирования и версионирования

Используйте систему Service Discovery

Команда разработки сервиса предоставляет и сопровождает интерфейсные библиотеки к своему сервису

Выделяйте части системы, которые могут пригодиться в других проектах, в отдельные сервисы и ведите их описание в каталоге микросервисов компании

croc_code

На связи в Telegram: @Ugway
email: alexey@patrin.ru

Telegram-канал Croc Code